DigibutlerK3

Documentation

22 octobre 2010

Sommaire

Historique :

Date	Version	Auteur	Description	
10 sept 2010	1.0	Henri Laidet	Création	

Présentation :

DigibutlerK3 est un projet à base de microcontrôleur 32bits permettant des applications domotique communicante (Ethernet, USB…).
C'est le grand frère de ***Digibutler*** (projet Elektor avril mai 2008).

Les outils logiciels sont gratuits (IDE, compilateur, Programmateur).

Spécifications techniques:

- Alimentation 8-15 Vdc
- Microcontrôleur MCF52259 (Kirin3)
  Coeur ColdFire V2, 80MHz, RAM 64k, Flash 512k, Mini FlexBus, USB OTG
- Horloge temps reel sauvegardée par pile.
- Ethernet sur RJ45
- 1 uart sur DE9
- 1 uart sur embase DIL
- 4 Ports d'IO sur embase DIL
- 1 sortie Relais
- 8 entrées analogiques sur embase DIL
- Support carte SD
- 512k SRAM
- Bus extension sur embase DIL
- QSPI sur embase DIL
- CanBus sur embase SIL
- USB host 2 connecteurs 1xUSB typeA et 1xMiniUSB
- PCB 120x95mm 4 couches

Schéma : ***« DIGI_K3_01A_sch.pdf »***
Liste des composants :***« DIGI_K3_01A_BOM.pdf »***
Sérigraphie : ***« DIGI_K3_01A_SER.pdf »***
Fichiers GERBER : ***« DIGI_K3_01A.zip »***

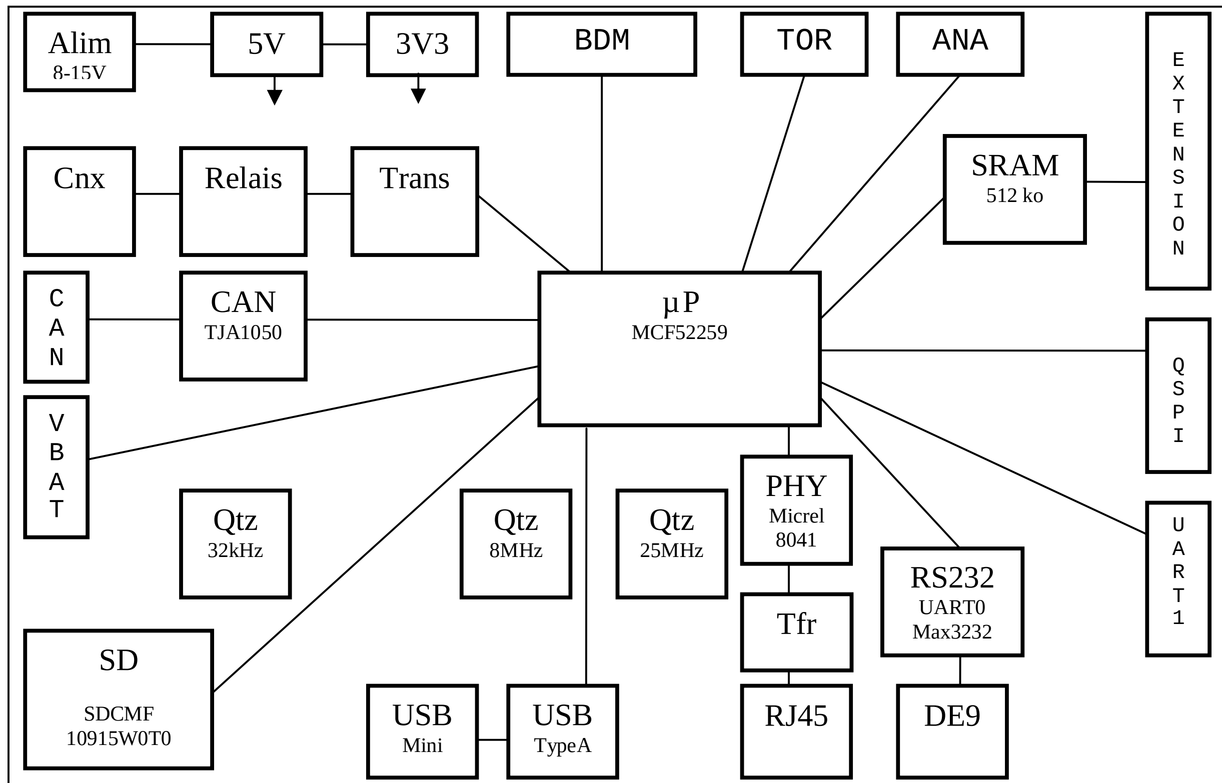


Figure 1

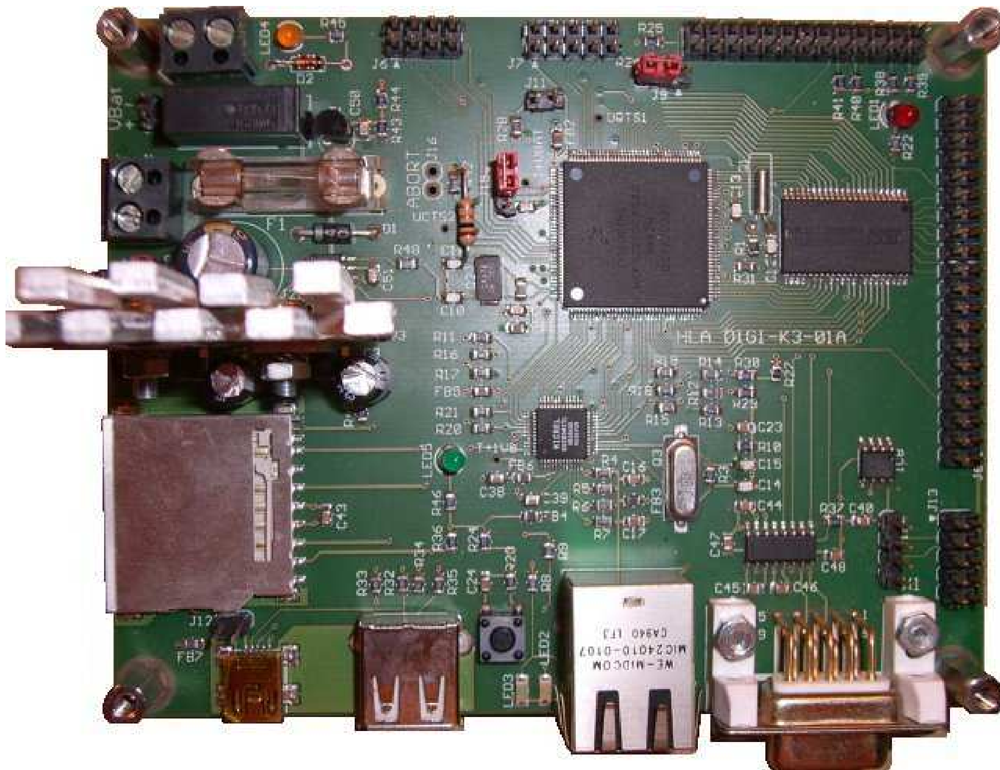


Figure 2

Description détaillée du matériel

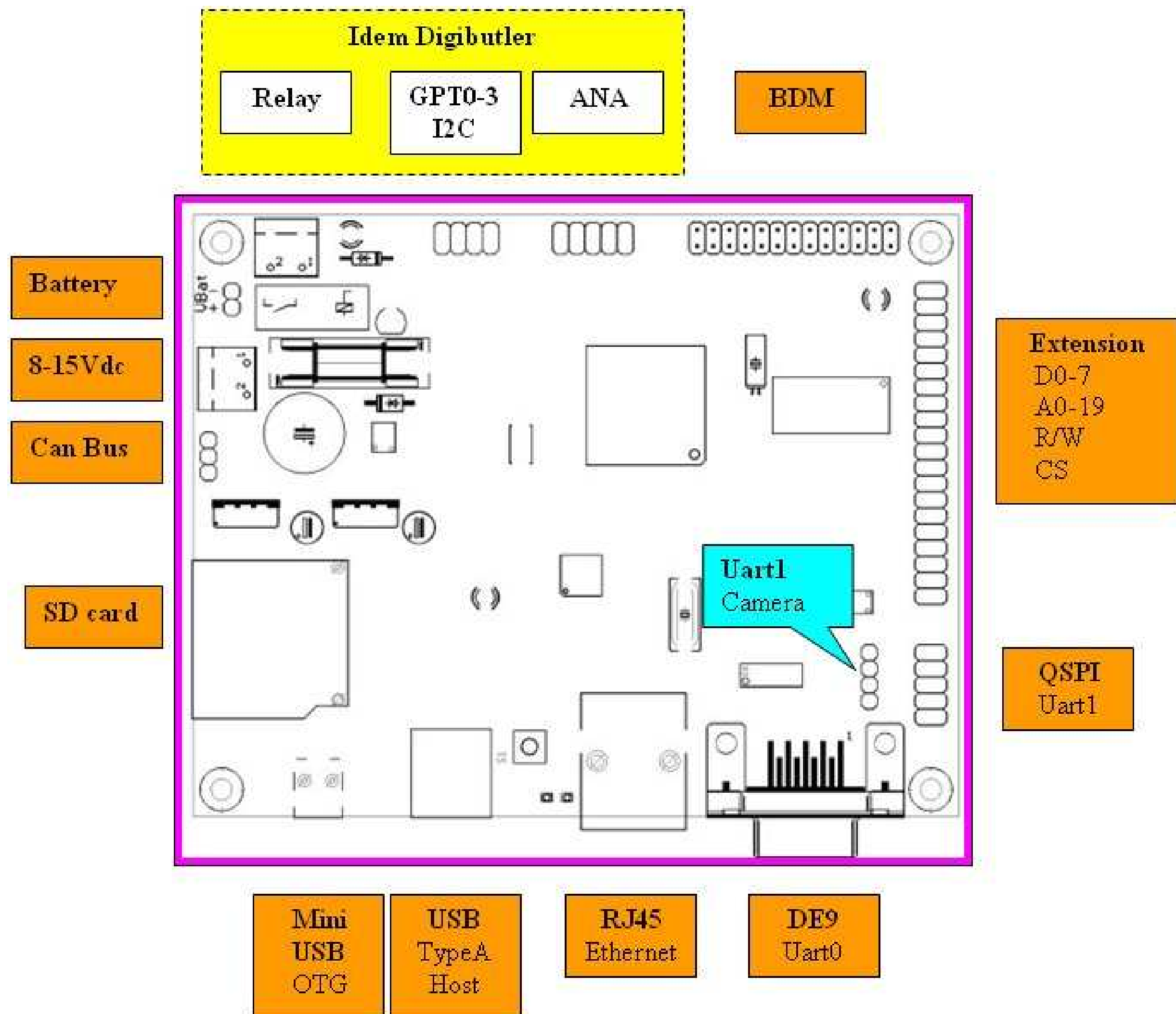


Figure 3

Alimentation

Entrée 8-15Vdc sur bornier Phoenix 2 voies au pas de 5.08mm
2 régulateurs en boitier TO220 produisent le +5V et le +3V3
- un 7805 pour le +5V USB et relais
- un LF33 pour le +3V3

Consommations (*adaptateur 9Vdc 500mA à l'entrée*):

1- moins de 300mA sans clé USB ni câble ethernet
2- moins de 350mA avec Ethernet link 100M
3- moins de 400mA avec link 100M + clé USB (2Go)

Le régulateur 5V doit dissiper moins de ¼ W
Le régulateur 3V3 doit dissiper environ 2.5 W
RthJC du LF33 = 3°C/W
Par sécurité, on prend TJ=100° et TA=40°
Un radiateur de 21°C/W est suffisant

Microcontrôleur

(Freescale) MCF52259CAG80 LQFP 144

Ethernet

Couche Physique :
(Micrel) KSZ8041TL boitier TQFP 48
RJ45 avec transfo intégré:
2 types possibles (double implantation)
- (Pulse) J0-0065NL avec LEDs intégrées
- (Wuerth Elek) MID-COM MIC24010-107T-LF3

RAM

(ISSI) IS61LV5128AL-10TLI 512 Ko 8bits 10ns boitier TSOP2 44

Carte SD

(MULTICOMP) SDCMF-10915W0T0
Connecté à DTIN0-3 pour une gestion en SPI software.
Le SPI hardware (QSPI) est conservé pour les contrôles SPI haut débit.

USB

2 connecteurs.
- Type A permet de connecter clé, clavier, souris (HID + Mass Storage)
- Mini USB permet une connexion OTG. Avec strapp d'alimentation Host ou Device

Liaisons séries

- Uart0 sur DE9 via un Max3232 (Rx, Tx, Rts, Cts).
- Uart1 en LVTTL sur J19 : embase SIL 4 voies (Rx, Tx, Gnd, +3V3)

SPI

QSPI sur J13 : embase DIL 10 voies (MISO, MOSI, SCK, CS0, Gnd, +3V3, +5V)
On peut connecter ici la carte d'extension "IO_EXP_01A" avec des expanders MicroChip.

GPTx et I2C

Ports d'entrées-sorties Tout Ou Rien GPT0-3 ainsi que I2C (SDA0 et SCL0) sur J6 : embase DIL 8 voies (mêmes signaux que Digibutler J6)

Analogique

Ports d'entrées Ana0-7 sur J7 : embase DIL 10 voies (mêmes signaux que Digibutler J7)

Relais

Relais OMRON G6D-1A sur J17 : bornier Phoenix 2 voies au pas de 5.08mm
Même montage du relais que Digibutler avec toutefois l'ajout d'un condo de 1µF sur la base du transistor qui permet de gérer un SPI software comme sur Digibutler sans commuter le relais.

BDM

Signaux du BDM coldfire sur J5 : embase DIL 26 voies
Utilisable avec TBLCF ou P&E …

BUS d'extension

J8 : Embase DIL 40 voies reçoit D0-7, A0-19, RW, OE, CS1 (CS0 est utilisé par la RAM)

Horloge temps réel

RTC du microcontrôleur.
Entrée VBat (2 piles AAA) sur J14 : embase SIL 2 voies
Vstby doit être compris entre 1V8 et 3V5
Istby = 5µA quand le 3V3 est présent
Istby = 25µA quand le 3V3 est absent

CAN

FlexCAN du microcontrôleur + TJA1050 sur J18 : embase SIL 3 voies (CanH, CanL, Gnd)
Un strapp connecte la résistance de terminaison de ligne 120 Ohms

Leds

Une Led présence +3V3
Une Led Reset
Une Led Relais fermé

Errata

Le schéma électronique comporte 2 erreurs affichées sur la dernière page du schéma.
Ces erreurs n'ont pas justifié de refaire le PCB.

1. Il manque une résistance de 10k entre le signal XTAL et le +3V3
2. la borne 5 du DE9 doit être mise au Gnd

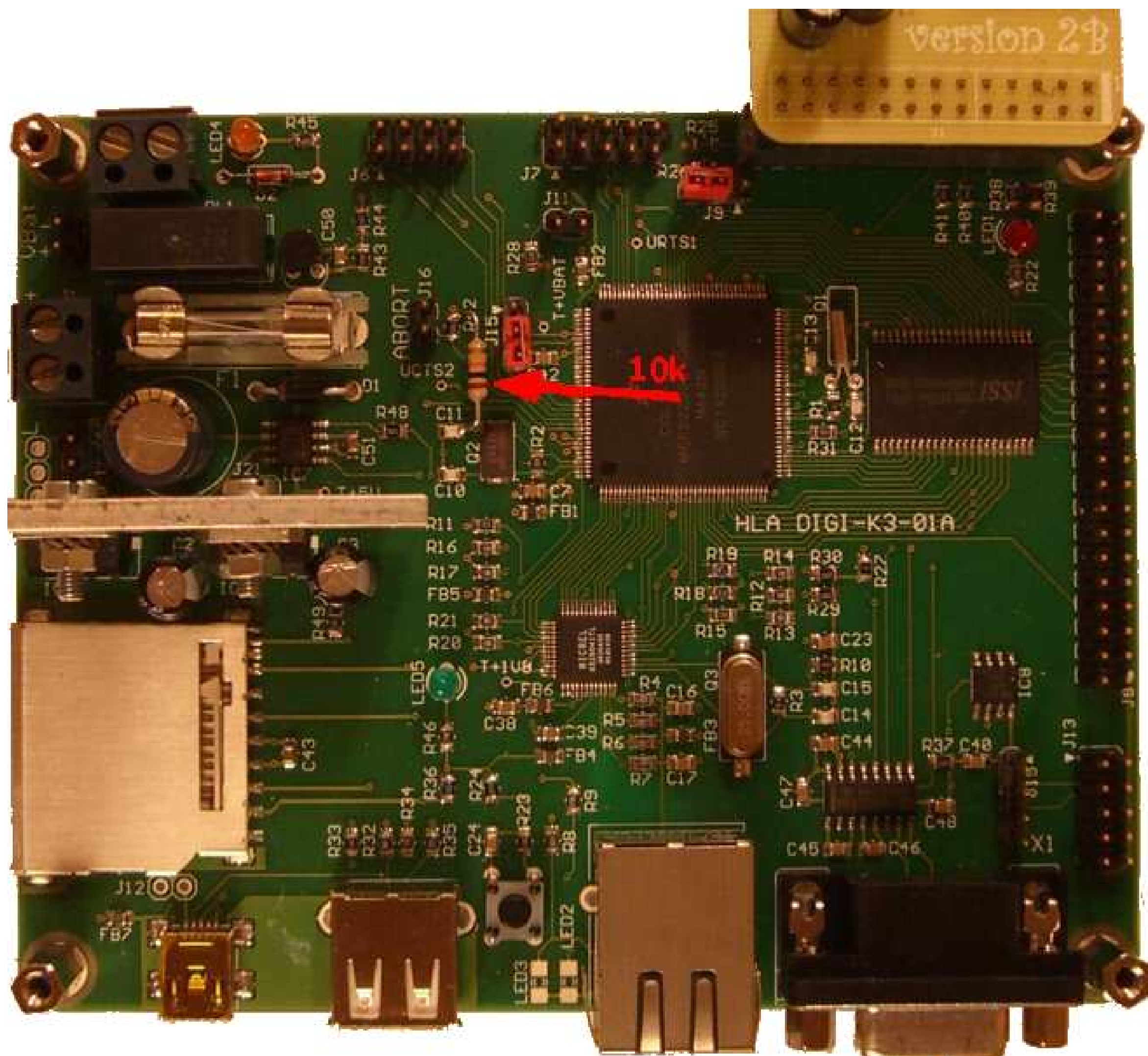


Figure 4

Connecteurs

J1 Alimentation

Pin	Nom	Description
1	+Vin	+8 à +15 Vdc
2	-Vin	0 V

J2 Support carte SD
J3 USB Type A
J4 MiniUSB

J5 BDM

J6 GPIO + I2C

Pin	Nom	Description
1	+3V3	
2	Gnd	
3	SDA0	I2C data Port AS1
4	SCL0	I2C clock Port AS0
5	GPT0	Port TA0
6	GPT1	Port TA1
7	GPT2	Port TA2
8	GPT3	Port TA3

J7 Entrées analogiques

Pin	Nom	Description
1	Gnd	
2	VDDA	Alimentation +3V3 du convertisseur analogique
3	AN1	Port AN1 entrée analogique
4	AN0	Port AN0
5	AN3	Port AN3
6	AN2	Port AN2
7	AN6	Port AN6
8	AN7	Port AN7
9	AN4	Port AN4
10	AN5	Port AN5

J8 Connecteur d'extension

Pin	Nom	Pin	Nom	Pin	Nom	Pin	Nom
1	+5V	11	A14	21	D6	31	A2
2	+5V	12	A15	22	D7	32	A3
3	+3V3	13	A12	23	D4	33	A4
4	+3V3	14	A13	24	D5	34	A5
5	Gnd	15	A10	25	D2	35	A6
6	Gnd	16	A11	26	D3	36	A7
7	A18	17	FB_CS1	27	D0	37	A8
8	A19	18	Gnd	28	D1	38	A9
9	A16	19	FB_RW	29	A0	39	Gnd
10	A17	20	FB_OE	30	A1	40	Gnd

J10 RJ45 Ethernet

J13 SPI Uart1

Pin	Nom	Description
1	+3V3	
2	Gnd	
3	UTXD1	Port UB0
4	URXD1	Port UB1
5	SPI_MOSI	Port QS0 QSPI MasterOut SlaveIn
6	SPI_CS0	Port QS3 QSPI SS/
7	SPI_MISO	Port QS1 QSPI MasterIn SlaveOut
8	SPI_SCK	Port QS2 QSPI SCK
9	+5V	
10	Gnd	

J14 Pile sauvegarde RTC

Pin	Nom	Description
1	VBat -	0 V
2	VBat +	+1V8 à +3V5 (2 piles AAA en série)

J17 Contacts du relais

J18 CAN

Pin	Nom	Description
1	CANH	Signal bus CAN High
2	Gnd	
3	CANL	Signal bus CAN Low

J19 Uart1

Pin	Nom	Description
1	+3V3	Tension d'alimentation du périphérique
2	UTXD1	Port UB0
3	URXD1	Port UB1
4	Gnd	

J21 VExt Report d'alimentation après fusible, diode d'inversion et condensateurs de filtrage

X1 DE9 Uart0 (console)

Pin	Nom	Description
1	CD	Pins 1, 4 et 6 reliées
2	TX	
3	RX	
4	DTR	Pins 1, 4 et 6 reliées
5	Gnd	
6	DSR	Pins 1, 4 et 6 reliées
7	CTS	
8	RTS	
9	nc	

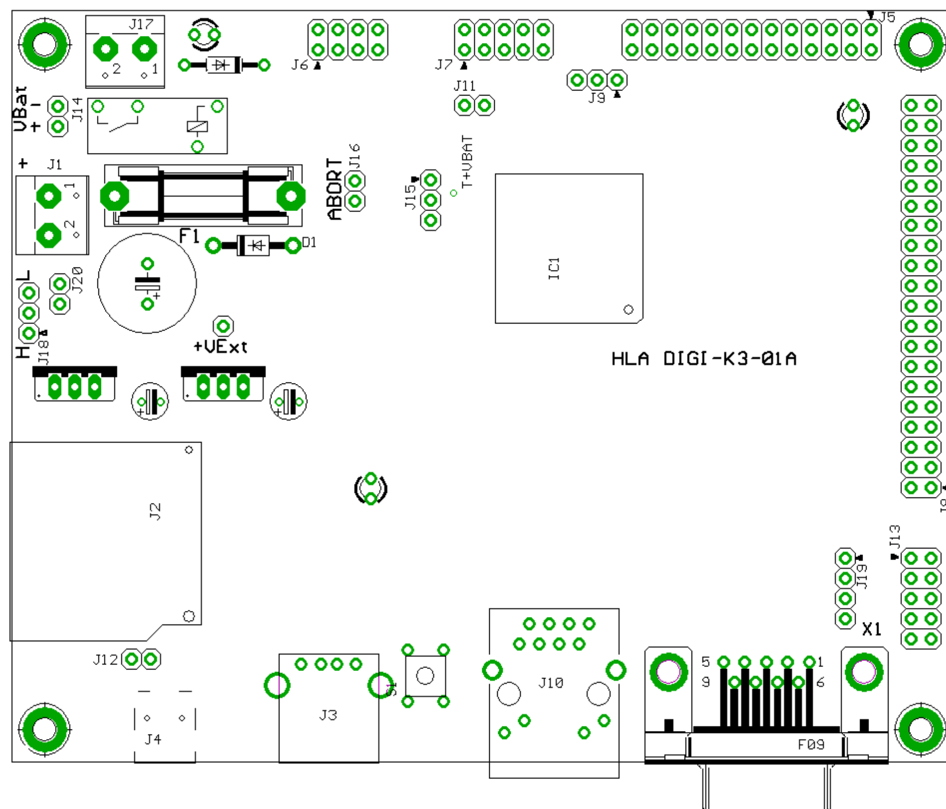


Figure 5

Strapps

J9 Sélection TCLK

Pin	Description	Position
1-2	TCLK sur pin 6 de J5	
2-3	TCLK sur pin 24 de J5 (recommandé datasheet Freescale)	oui

J11 Sélection BDM ou JTAG

Strapp	Description	Position
Présent	Mode JTAG	
Absent	Mode BDM (Debug module)	oui

J12 Alimentation MiniUSB

Strapp	Description	Position
Présent	Mini USB en mode Host (+5V pin 1 pour alimenter le périphérique)	
Absent	Mini USB en mode Device	oui

J15 Sélection RTC avec ou sans pile

Pin	Description	Position
1-2	RTC sauvegardée par une pile sur VBat (J14)	oui
2-3	RTC non sauvegardée. Utilisation du +3V3 de la carte	

J16 Abort sur IRQ7

J20 Terminaison CAN

Strapp	Description	Position
Présent	Résistance fin de ligne bus CAN	oui
Absent	Pas de résistance	

Outils logiciels

Chaîne de compilation gratuite Eclipse + GCC

IDE Eclipse

Téléchargement

IDE Eclipse à télécharger depuis http://www.eclipse.org/downloads/
Choisir : ***Eclipse IDE for C/C++ developers***
Pour la dernière version (aujourd'hui helios 3.6) ***"eclipse-cpp-helios-win32.zip"***
Ou choisir : ***Older versions*** pour (galileo) ***"eclipse-cpp-galileo-SR1-win32.zip"***

Installation

Il suffit de dézipper le fichier dans le répertoire choisi.
L'exécutable « eclipse.exe » se trouve dans le répertoire d'installation

Compilateur assembleur linkeur

Téléchargement

Compilateur GCC de CodeSourcery
A télécharger depuis http://www.codesourcery.com/

1- Sourcery G++
2- EDITIONS->LITE
3- ColdFire http://www.codesourcery.com/sgpp/lite/coldfire
4- **Download** the current release
5- ELF ***"freescale-coldfire-4.4-215-m68k-elf.exe"***

Installation

Le fichier téléchargé est un exécutable qui installe le cross compilateur **"m68k-elf-gcc"**.
Nous n'avons pas la nécessité d'accéder au répertoire d'installation sauf peut-être pour lire la doc de GCC (compilateur, assembleur, linkeur, archiveur) ainsi que des différents utilitaires.

Logiciel

Le logiciel de ***DigibutlerK3*** est basé sur **"Freescale MQX RTOS"** version3.51.
La licence est **"FREESCALE SEMICONDUCTOR SOFTWARE LICENSE AGREEMENT"**.
Fichier ***"license.txt"*** dans le projet ***"DigiK3_MQX"*** (utilisation libre).

La base MQX est composée de :

1. Freescale MQX RTOS (Real Time OS)
2. Freescale MQX RTCS (Communication Suite)
3. Freescale MQX File System (MFS)
4. Freescale MQX USB Host/Device Stack

Le code source de la totalité du projet est présent dans ***"ExportKirin3 AAAA MM JJ.zip".***

Importation du projet ***"ExportKirin3 AAAA MM JJ.zip"*** dans l'environnement de travail.

1- Lancer l'IDE « eclipse.exe »
2- Créer un workspace vide : ***File->Switch Workspace->Other***
3- Importer ***File->Import...***
4- Choisir ***General->Existing Projects into Workspace*** puis ***Next***
5- Cocher ***Select archive file*** puis ***Browse… "ExportKirin3 AAAA MM JJ.zip"***
6- Cliquer ***Select All*** puis ***Finish***

Tous les projets de l'archive se retrouvent dans le workspace

1.	**DigiK3_MQX**	Projet application	
2.	**LIBRARY**	Librairie ***"libHLA.a"*** contenant toutes les librairies qui suivent	
3.	**MFS_LIB**	Librairie ***"libMFS_LIB.a"***	file system
4.	**MQX_BSP_LIB**	Librairie ***"libMQX_BSP_LIB.a"***	board support package
5.	**MQX__LIB**	Librairie ***"libMQX_LIB.a"***	kernel générique
6.	**MQX_PSP_LIB**	Librairie ***"libMQX_PSP_LIB.a"***	plateforme support package
7.	**RTCS_LIB**	Librairie ***"libRTCS_LIB.a"***	stack TCP/IP
8.	**SHELL_LIB**	Librairie ***"libSHELL_LIB.a"***	commandes shell
9.	**USB_LIB**	Librairie ***"libUSB_LIB.a"***	stack USB host et device

Une fois importé ***"ExportKirin3 AAAA MM JJ.zip"***, tous les projets (librairies + application) sont compilés automatiquement.
La première compilation est assez longue (ex : P4 3GHz 2Go RAM XP => 20 minutes).
Ensuite, il y a seulement ***"DigiK3_MQX"*** à compiler.

Le projet application est ***"DigiK3_MQX"***.
La librairie ***"libHLA.a"***. dans ***"LIBRARY"*** est la concaténation de toutes les autres librairies.
L'utilisation de plusieurs librairies générait des erreurs lors du link ***m68k-elf-ld*** (les appels croisés ne sont pas tous résolus).
Il y a plusieurs librairies car l'archiveur ***m68k-elf-ar*** oublie des fichiers à archiver si il y en a trop (commande trop longue **tronquée** : sûrement une variable d'environnement à modifier).

Configuration du projet "DigiK3_MQX"

Le fichier ***"defines.h"*** configure l'application avec un pack de #define

APPCFG_ENABLE_USB_FILESYSTEM

Activation de la prise en charge d'une clé USB [c:]
Sources dans répertoire ***USB***

APPCFG_ENABLE_RTCS

Activation de la stack TCP/IP
Nous avons le choix avec ou sans DHCP

- avec DHCP

1. activer APPCFG_ENABLE_DHCP
2. fixer la MAC de la carte dans RTCS.c

- sans DHCP

1. désactiver APPCFG_ENABLE_DHCP
2. fixer MAC, IP, MASK et GATEWAY dans RTCS.c

Source : ***RTCS\RTCS.c***

APPCFG_ENABLE_DHCP

Activation client DHCP

APPCFG_ENABLE_WEBSERVER

Activation du serveur http
Sources dans la librairie ***RTCS_LIB\httpd***

APPCFG_ENABLE_HTTP_TUNNEL_FTP

Activation d'une requète supplémentaire dans serveur http permettant :

- lecture et écriture de fichiers sur la carte SD
- dialogue avec PLC (voir ***APPCFG_ENABLE_PLC***)

Source : ***RTCS\http_tunnel_ftp.c***

APPCFG_ENABLE_SD_CARD

Activation prise en charge carte SD [a :]
Source : ***SD_CARD\SD_Task.c***
Utilise le driver ***"sspi0:"*** qui gère le SPI software de la carte SD

APPCFG_ENABLE_RAMDISK

Activation d'un Disque RAM [b:] de 128ko
Nécessaire si APPCFG_ENABLE_HTTP_TUNNEL_FTP est activé car génération de fichiers d'échange.
Source : ***source\DigiK3_Ramdisk.c***

APPCFG_ENABLE_IO_DIGITAL
APPCFG_ENABLE_IO_ANALOG
APPCFG_ENABLE_IO_EXPANDERS

Activation process d'entrées sorties Tout Ou Rien et Analogique
Un ***Timer_Callback*** est programmé à 10 ms.
A chaque ***Timer_Callback***, les entrées sorties sont mises à jour.

APPCFG_ENABLE_IO_DIGITAL active la gestion des IO TOR.
Les ports sont seulement initialisés avec « `fopen("gpio:read" et "gpio:write"` »
L'application (PLC) lit et écrit directement les ports.

APPCFG_ENABLE_IO_ANALOG active la gestion des entrées analogiques.
Le convertisseur est programmé puis à chaque mise à jour, les 8 voies analogiques sont lues et stockées dans les variables : «« `short DATA_Analog[8] ;` »»
L'application dispose des valeurs analogiques.

APPCFG_ENABLE_IO_EXPANDERS active la gestion d'une carte d'extension d'entrées sorties TOR " IO_EXP_01A" connectée sur J13 (QSPI)
Carte où sont montés 2 expanders MCP23S17 de Microchip.
Le QSPI est programmé à 10MHz qui est la vitesse maxi des expanders MCP23S17.
Les expanders de la carte d'extension sont programmés (un en entrée et l'autre en sortie).
A chaque mise à jour, les données des expanders sont stockées ou lues dans les variables : ## uint_16 IO_exp_data[NB_IO_EXPANDERS]## en respectant la direction de chaque bit de port.
L'application n'a plus qu'à lire dans IO_exp_data[0] et écrire dans IO_exp_data[1].
La mise à jour d'une carte « IO_EXP_01A » contenant 2 expanders dure environ 10 micro secondes.

Source : ***source\DigiK3_IO.c***

APPCFG_ENABLE_CAMERA

Activation caméra C328R connectée sur J19 (uart1)
L’image est capturée sans arrêt et est stockée dans un buffer en RAM externe
- soit dans un fichier BMP si on active PREVIEW_MODE dans Camera.h
- soit dans un fichier JPG si on désactive PREVIEW_MODE
Le programme installe TFS pour accéder aux fichiers dans la RAM externe
On consulte l’image avec une page WEB en utilisant une balise :
<img src="TFS/MCF5225X.JPG"> ou <img src="TFS/MCF5225X.BMP">
Le serveur http a un mutex pour le fichier JPG.

Source : ***Camera\Camera_Task.c*** code
Camera\Camera_Vars.S déclaration des buffers en RAM externe

APPCFG_ENABLE_PLC

Activation du PLC (Programmable Logic Controller)
C’est un petit automate (idem Digibutler) programmable en langage à contact avec un Editeur graphique ***« H_DigiPLC »*** annexe.
Le moteur PLC est appelé par ***Timer_Callback*** des entrées sorties toutes les 10ms.
La communication avec l’éditeur se fait via le tunnel FTP du serveur HTTP (transparent pour l’utilisateur)

Source : ***PLC\PLC_Task.c*** code
PLC\PLC_Vars.c déclaration des buffers en RAM externe

APPCFG_ENABLE_CAN

Activation serveur de monitoring CAN bus
Le FlexCAN est initialisé pour recevoir les messages CAN pour tous les identifiers.
Tous les messages reçus sont datés à la micro seconde et placés dans un FIFO.
Un serveur se met à l’écoute sur le port 1233 et attend la connexion du client ***“H_CANdiag”*** annexe.

Source: ***CAN\CAN_Task.c*** code
CAN\CAN_Vars.S declaration Fifo en RAM externe

APPCFG_ENABLE_RTCS_CONFIG_FILE

Activation lecture de la configuration dans le fichier ***« DigiK3.cfg »*** sur la carte SD.
Permet de changer MAC, IP, MASK, GATEWAY
Chaque ligne du fichier doit contenir une commande d'affectation.
Les mots clés sont :

#MAC: suivi de 6 valeurs numériques décimal ou hexa (préfixe 0x ou $).
le séparateur est le point ou la virgule indifféremment.
Exemple : #MAC:0x00,0xcf,0x52,0x25,0x9F,0xec

#IP : suivi de 4 valeurs décimal ou hexa
Exemple : #IP:192.168.1.151

#MASK : suivi de 4 valeurs décimal ou hexa
Exemple : #MASK:255.255.255.0

#GATEWAY: suivi de 4 valeurs décimal ou hexa
Exemple : #GATEWAY:192.168.1.1

La lecture du fichier de configuration est dans la tache RTCS avant de programmer le Fast Ethernet Controller (FEC).

1 - Les valeurs par défaut sont chargées dans les variables
2 – Attente 800ms que la carte SD soit opérationnelle.
Il faut environ 550ms après un reset pour que la carte SD soit OK
3 – Appel de ***« ReadConfigFile »*** qui se trouve dans ***« DigiK3_Tools.c »***
« ReadConfigFile » décode ***« DigiK3.cfg »*** si il est présent et surcharge les Variables.
4 – Programmation du FEC
5 – Initialisation des taches qui ont besoin de la stack TCP/IP (HTTP, serveur CAN…)

Script de link

Le fichier « ***intflash.lcf*** » décrit les sections et leurs adresses

Dans la partie ***MEMORY***, on doit fixer l'origine de l'application dans la Flash ou dans la RAM externe.

```
----- Flash ---------------------------------------------
rom        (RX): ORIGIN = 0x00000000, LENGTH = 0x00080000
----- RAM -----------------------------------------------
rom        (RX): ORIGIN = 0x10000000, LENGTH = 0x00080000
```

Application en Flash

Il y a uniquement l'origine de ***rom*** qui doit être égale à 0x00000000.
Il n'y a rien de plus à configurer car le ColdFire amorce en Flash.

Application en RAM externe

Il faut que l'origine de ***rom*** soit égale à 0x10000000.
Le ColdFire amorçant dans la Flash, il est nécessaire de Flasher au moins une fois l'application pour que le ColdFire puisse amorcer.

Au démarrage (en Flash), après avoir initialisé le FlexBus, l'application contrôle si il y a du code valide dans la RAM (identificateur 12 octets + checksum).
Si le code en RAM est valide (l'application linkée en 0x10000000), c'est lui qui est exécuté et non celui qui est en Flash.
Cela permet de déboguer plus facilement avec des points d'arrêt soft (0x4AFC).

Le code en RAM externe est toutefois 10 fois plus lent qu'en Flash.
- SRAM 10ns 8 bits + FlexBus à 80MHz + 1 WaitState (à 80MHz 1 waitstate est obligatoire)
- accès bus 8 bits = S0 + S1 + SW + S2 + S3 = 5*12.5ns = 62.5ns
- accès bus 32 bits = 4*62.5ns = 250ns

Compilation

Si ***Project->Build Automatically*** est coché toute modification/sauvegarde d'un fichier lance la compilation du projet en compilant uniquement les fichiers nécessaires.

Il y a toutefois une anomalie dans le contrôle des dépendances avec les fichiers assembleur (*.S). Les fichiers assembleur **ne sont pas assemblés** si il y a une modification d'un #define qui les concerne.
Exemple : Avec APPCFG_ENABLE_PLC=1, si on modifie PLC_BUFFER_SIZE alors PLC_Vars.S n'est pas réassemblé. Il y a de ce fait une incohérence entre la section en « C » et la section en assembleur.

Les fichiers assembleur du projet sont :
- Camera_Vars.S
- PLC_Vars.S
- CAN_Vars.S

Ils servent à placer les buffers dans la RAM externe.
Techniquement, pour attribuer une section particulière aux buffers sans utiliser « **`__attribute__ ((section(".mysection")))`**» qui génère dans le fichier ELF une zone remplie de zéros.

Conclusion : Si on modifie un #define qui peut influencer le contenu des *.S alors il faut relancer la compilation à la main : ***Projet->Clean…Clean projects selected below*** et sélectionner **DigiK3_MQX** ce lancera la génération complète du projet (mais pas des librairies).

Programmation Flash ou RAM

La programmation de l’application dans la Flash de ***DigibutlerK3*** se fait avec :

- Matériel ***TBLCF*** connecté au BDM idem ***Digibutler***
- Logiciel ***H_Flasher*** non limité en taille de code.

H_Flasher utilise les fichiers XML de CodeWarrior V6.x ou 7.x pour connaître les caractéristiques du ColdFire :

- Taille et organisation de la Flash
- Taille de la RAM interne
- …

Au premier lancement de H_Flasher, on doit renseigner les champs de configuration :
Les champs 1, 2 et 3 permettent l’accès aux fichiers XML de CodeWarrior.
Le champ 4 définit la plateforme utilisée (« MCF52259_INTFLASH.xml » pour DigibutlerK3).
Le champ 5 définit le hardware utilisé.

- ***TBLCF*** nécessite ***tblcf.dll*** accessible par ***H_Flasher***
- ***OSBDM*** nécessite ***osbdm_cf2.dll*** . Il fonctionne avec la carte d’évaluation M52259DEMO qui a un OSBDM première génération intégré

Ensuite, on doit définir le fichier application à programmer.
Ce fichier peut être indifféremment au format ELF ou SRecord.

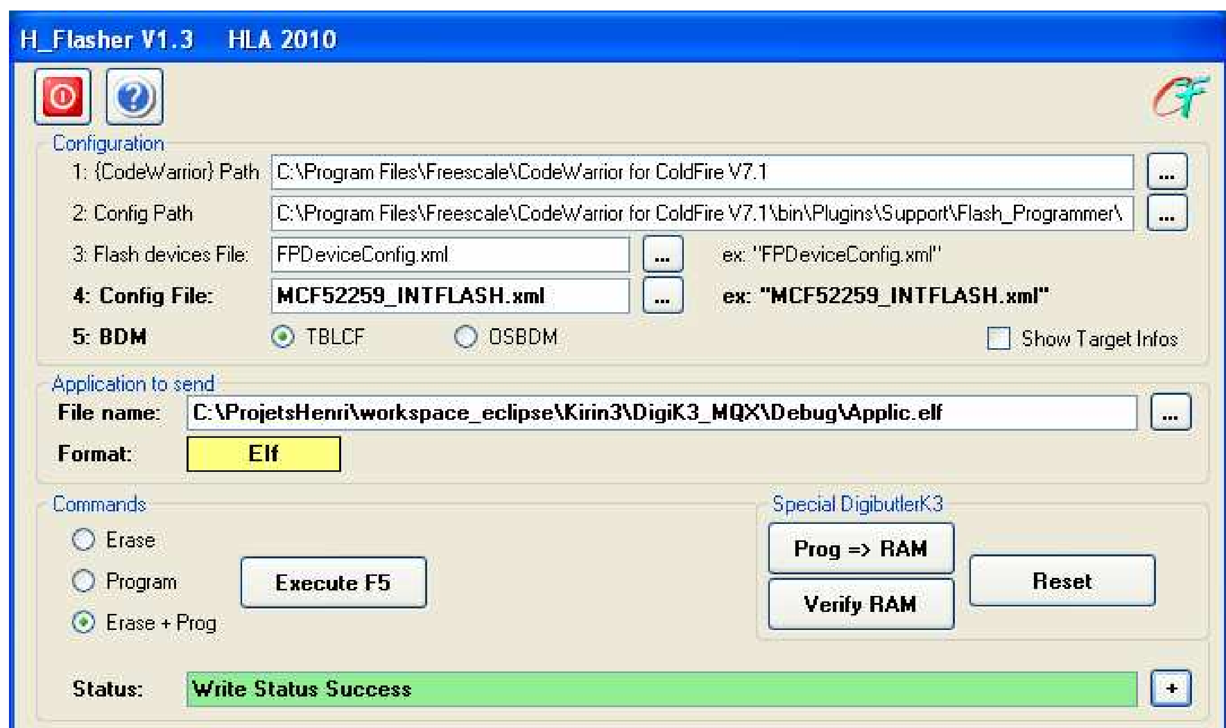


Figure 6

Toute cette configuration est sauvegardée dans le fichier ***H_FlashSettings.xml*** et relu au prochain lancement.

Les commandes Flash sont Erase, Program ou Erase+Prog.
Les commandes RAM sont spécifiques à ***DigibutlerK3*** et ne peuvent pas être utilisées sur d'autres plateformes.

H_Flasher peut rester ouvert entre plusieurs utilisations.
Le fichier application est rechargé à chaque programmation Flash ou RAM.

H_Flasher peut être utilisé avec plusieurs ColdFire : MCF52231 (Digibutler), MCF52233, MCF52259, MCF5208.

Projet : ***H_Flasher*** est écrit avec Microsoft Visual C++ 2008
Sources jointes : ***« H_Flasher AAAA MM JJ.zip »***

Annexes

Shell

Lorsque l'application est opérationnelle, nous avons un petit shell qui propose quelques commandes bien utiles :

```
shell> help
Available commands:
   netstat
   ipconfig [<device>] [<command>]
   cd       <directory>
   copy     <source> <dest>
   date     [<year>] [<month>] [<day>]
   del      <file>
   dir      [<filespec>] [<attr>]]
   format   <drive:> [<volume label>]
   help     [<command>]
   mkdir    <directory>
   pwd
   ren      <oldname> <newname>
   rmdir    <directory>
   sys      [stack | lwmemblock]
   time     [<hour>] [<minute>] [<second>]
   type     <filename>
   ?
   xfb      <address> <bytes> <value>
   xmd      <address> [<bytes>]
   xvt      [<init>]
```

`netstat` : affichage les statistiques IP, ICMP, UDP et TCP

`ipconfig` : affichage ou modification IP, MASK, GATEWAY

`cd` : change le répertoire

`copy` : copie de fichier

`date` : affichage ou modification de la date

`del` : effacement fichier

`dir` : affichage liste des fichiers du répertoire actif

`format` : formatage disque (carte SD, clé USB)

`mkdir` : création répertoire

`pwd` : affichage du répertoire actif (son nom mais pas son contenu)

`ren` : renomme un fichier

`rmdir` : effacement d'un répertoire

`sys` : affichage de l'état système = Etat des piles des taches + RAM des processus

`time` : affichage ou modification de l'heure

`type` : affichage d'un fichier texte

`xfb` : remplissage d'une zone de RAM (fill block)

`xmd` : affichage d'une zone de RAM (memory display

`xvt` : affichage du nombre d'interruptions de tous les vecteurs (debug)

H_DigiPLC

C'est un utilitaire sur PC qui se connecte à ***DigibutlerK3*** sur le port HTTP avec une requête de Tunnelling FTP.
Pour que la connexion soit possible, il faut activer 2 #defines **APPCFG_ENABLE_WEBSERVER** et **APPCFG_ENABLE_HTTP_TUNNEL_FTP**.

Il permet de transférer des fichiers entre PC et ***DigibutlerK3***.
Il sert aussi d'éditeur PLC quand le #define **APPCFG_ENABLE_PLC** est activé.

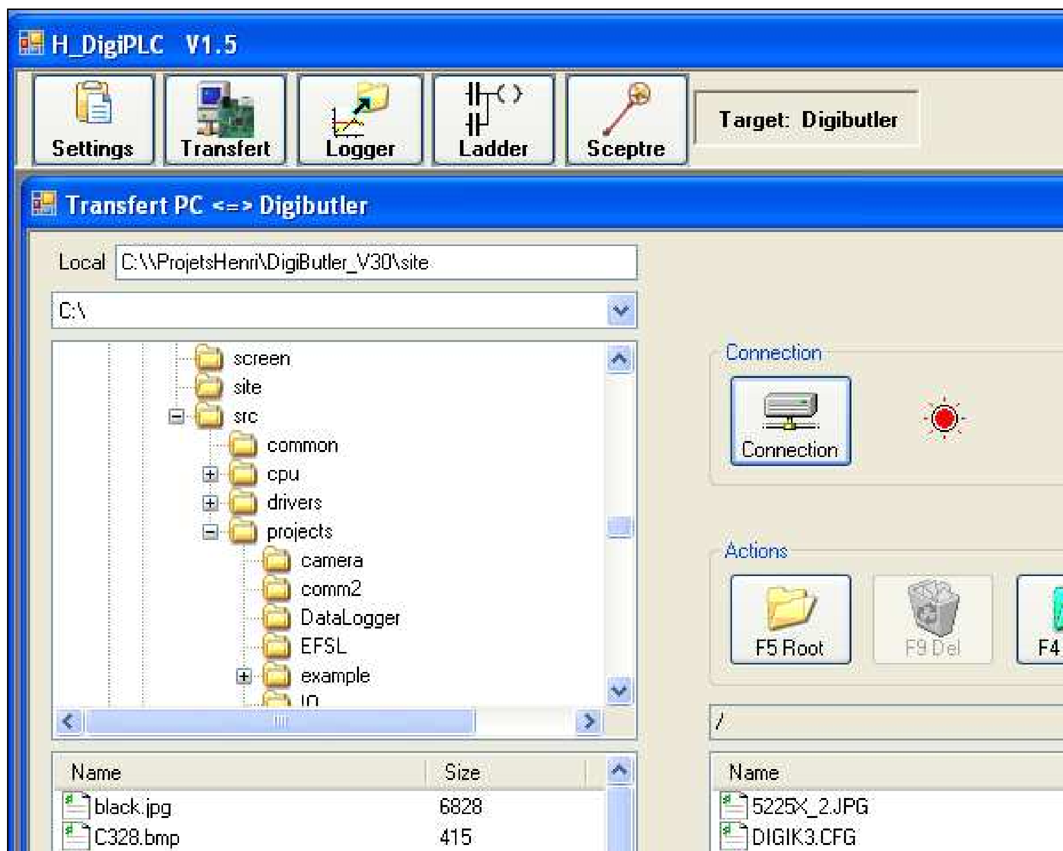


Figure 7

Projet : ***H_DigiPLC*** est écrit avec Microsoft Visual C# 2008
Sources jointes : ***« H_DigiPLC AAAA MM JJ.zip »***
Documentation : ***« H_DigiPLC_fr.pdf »***

H_CANdiag

C'est un utilitaire sur PC qui se connecte à ***DigibutlerK3*** sur le port 1233
Pour que la connexion soit possible, il faut activer le #define **APPCFG_ENABLE_CAN**

Il permet de faire du monitoring de bus CAN.

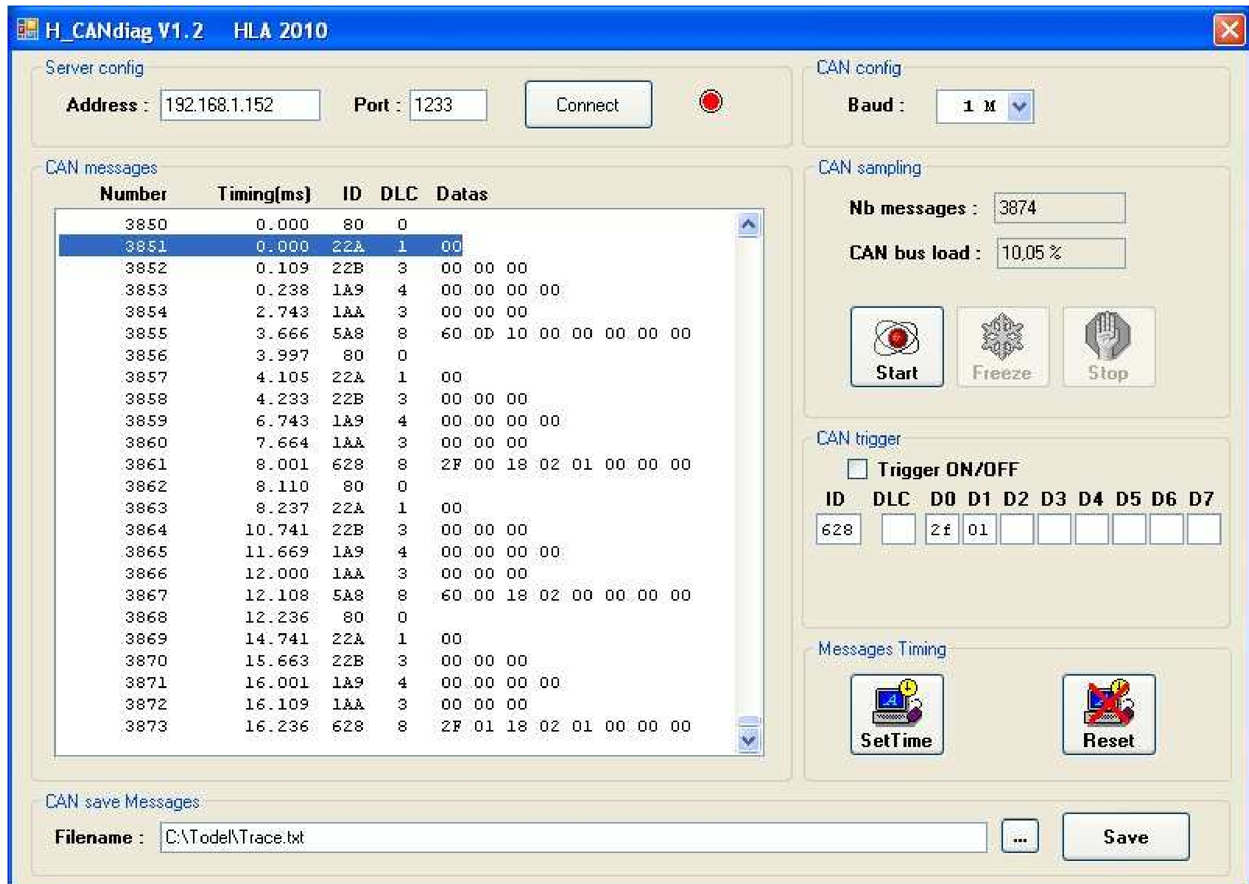


Figure 8

Projet : ***H_CANdiag*** est écrit avec Microsoft Visual C++ 2008
Sources jointes : ***« H_CANdiag AAAA MM JJ.zip »***
Documentation : ***« H_CANdiag_fr.pdf »***

IO_EXP_01A

C’est une carte d’extension d’entrées sorties.
Alimentation +5V à +8Vdc
16 entrées TOR 24Vdc
8 sorties relais
8 sorties LVTTL (ports MCP23S17)
Pilotage SPI
PCB double face

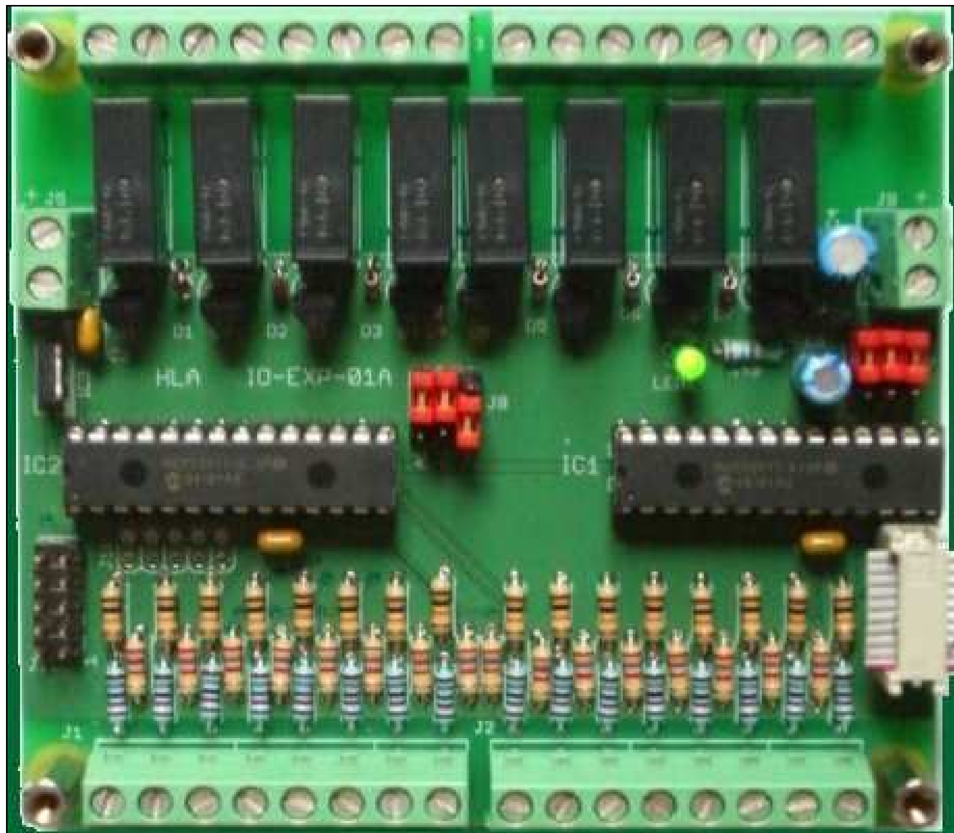


Figure 9

Schéma : *« IO_EXP_01A_sch.pdf »*
Liste des composants :*« IO_EXP_01A_BOM.pdf »*
Sérigraphie : *« IO_EXP_01A_SER.pdf »*
Montage : *« IO_EXP_01A_montage.pdf »*
Fichiers GERBER : *« IO_EXP01A.zip »*
Documentation : *«A_lire.txt »*

Liste des fichiers joints

"DigibutlerK3.zip"	
"DIGI_K3_01A_sch.pdf "	Schéma électronique
"DIGI_K3_01A_BOM.pdf "	Liste des composants
"DIGI_K3_01A_SER.pdf "	Sérigraphie
"DIGI_K3_01A.zip "	Fichiers de fabrication (GERBER, EXCELLON)
"ExportKirin3 AAAA MM JJ.zip"	Sources application ***DigibutlerK3***
"H_Flasher AAAA MM JJ.zip"	Sources programmateur de Flash MS VC++ 2008
"H_DigiPLC.zip"	
"H_DigiPLC AAAA MM JJ.zip"	Sources éditeur PLC MS VC# 2008
"H_DigiPLC_fr.pdf "	Documentation
"H_CANdiag.zip"	
"H_CANdiag AAAA MM JJ.zip"	Sources monitoring CAN MS VC++ 2008
"H_CANdiag_fr.pdf "	Documentation
"IO_EXP_01A.zip"	
" IO_EXP_01A_sch.pdf "	Schéma électronique
"IO_EXP_01A_BOM.pdf "	Liste des composants
"IO_EXP_01A_SER.pdf "	Sérigraphie
"IO_EXP_01A_montage.pdf "	Montage
"IO_EXP01A.zip "	Fichiers de fabrication (GERBER, EXCELLON)
"A_lire.txt "	Documentation